BART
(B AUTOMATIC REFINEMENT TOOL)
-
GUI USER MANUAL

Réf. :	Version : 1.0	Date :

CLEARSY Société par Actions Simplifiée au Capital de 266 880 Euros. - RCS Aix-en-Provence 433 901 402 - Code NAF 721 Z
320, Avenue Archimède – Les Pléiades III - Bât A - 13857 AIX EN PROVENCE CEDEX 3
Tél : 04 42 37 12 70 – Fax : 04 42 37 12 71

Revisions

Version	Date	Comment
1.0		Initial version

Ce document est la propriété de ClearSy - Toute reproduction ou utilisation partielle ou totale de celui-ci est interdite sans son accord préalable.

REFERENCES

Reference	Title	Version
[R1]	BART – User Manual	1.0

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

INDEX

Revisions....2
References....3
Index....4
I Introduction....6
II Usage....7
II.1 Using Bart GUI from AtelierB....7
II.2 Advanced users - From the command line....7
II.2.1 Command usage....8
II.2.2 Input files....8
II.2.3 Visibility for loaded component....8
II.3 Advanced users – Selecting a new component to refine directly in the Bart GUI....9
III GUI Overview....10
III.1 Menu bar....10
III.1.1 File menu....11
III.1.2 Edit menu....11
III.1.3 Rule files menu....11
III.1.4 Refinement menu....11
III.1.5 Conflicts menu....11
III.2 Action bars....12
III.2.1 Refinement control buttons....12
III.2.2 Zoom buttons....12
III.2.3 Refresh button....13
III.2.4 Stop button....13
III.3 Variables tab....13
III.4 Operations tab....14
III.5 Rule files tab....14
IV GUI settings....15
IV.1 Environment directories....15
IV.2 Preferences....15
IV.2.1 Commands....15
IV.2.2 Rule files....16
IV.2.3 Directories....16
IV.2.4 Display....16
V Method of interactive refinement....17
V.1.1 When is interactive refinement necessary?....17
V.1.2 Interactive variable refinement....17
V.1.3 Interactive operation refinement....18
VI Interactive refinement of variables....19
VI.1 Consulting found variable rules - Variables rule tree....19

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VI.2 Displaying a selected rule – Variable rule display panel.....20
VI.3 Consulting hypothesis for refinement - Hypothesis panel.....21
VI.3.1 Filtering hypothesis by type – Filters menu.....21
VI.3.2 Filtering hypothesis by content – Filter text bar.....22
VII Interactive refinement of operations – Operations tab.....24
VII.1 Selecting an operation - Operation list view.....24
VII.2 Current refinement status – Refinement status bar.....25
VII.2.1 Operation name.....25
VII.2.2 Refinement type.....25
VII.2.3 Refinement status.....26
VII.3 Processing and navigating in an operation refinement.....26
VII.3.1 Bart GUI refinement tree.....26
VII.3.2 Consulting the refinement tree - Refinement tree view.....27
VII.3.3 Consulting the refinement result – Refinement result panel.....30
VII.3.4 Node depending tabs.....32
VII.3.5 Zooming on tree parts.....34
VIII Adding and editing RMF files – Rule files tab.....37
VIII.1 Managing rule files – Rule files list view.....37
VIII.2 Listing rule file content – Theories tree.....38
VIII.3 Consulting rule file content – RMF display tab.....39
VIII.4 Handling rule file errors – Error view.....39
VIII.5 Rule file modifications during interactive refinement.....40
IX Deadlock resolution with Bart GUI.....41
IX.1 Usage with automatic refinement.....41
IX.2 Getting deadlock information – Conflicts window.....41
Appendix A: Table of actions.....44
Appendix B: Table of figures.....45

Ce document est la propriété de ClearSy - Toute reproduction ou utilisation partielle ou totale de celui-ci est interdite sans son accord préalable.

I INTRODUCTION

The aim of this document is to introduce the concept of graphical refinement, through explanations of usage and functionalities of Bart graphical user interface (GUI).

It will help the user to determine in which cases the Bart GUI must be used, and how it can be processed aside with classic Bart automatic refinement.

Interactive refinement functionalities will be presented and ordered according to graphical component hierarchy. All along the document, summary tables of actions will be presented. All these actions are gathered in actions table, at the end of this document. This table can be used to easily navigate through this document for searching certain functionalities.

The Bart GUI doesn't allow processing the complete refinement, as output result components are not generated. It is used in the case issues were encountered during automatic refinement. So the GUI could be seen as a debug tool for the automatic refinement process. It allows the user to see which rules have been chosen for refinement, and, if so, which variables or operations couldn't be refined.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

II USAGE

This section describes different ways to launch interactive refinement with the Bart GUI.

Note: For all launching methods, Bart GUI is supposed to work on type-checkable components. So user should ensures, if it is not automatically done (as with AtelierB, which does not allow launching refinement on a component that is not type-checked), that components are correct.

II.1 Using Bart GUI from AtelierB

As for automatic refinement (cf. [R1]), the usual way for processing graphical interactive refinement is to launch the Bart GUI from AtelierB 4 interface. Unlike automatic refinement, and as said in introduction, interactive refinement will not generate output components and so will not add any components to project component list.

Interactive refinement can be simply launched in AtelierB by selecting a component and choosing "Interactive refinement" in "Component" menu. AtelierB then launches the Bart GUI with suitable values for the parameters described in II.2 :

- Automatic selection of directories containing the environment of the component to refine (seen machines and abstraction)
- Automatic selection of Bart rule file associated to the component if any (file with same name as the component and the .rmf extension, which must be present in source directory)
- Automatic selection of Bart predefined rule base, which comes with AtelierB (file PatchRaffiner.rmf)

Parameters provided by AtelierB to the GUI can be modified or completed later, directly in the GUI.

II.2 Advanced users - From the command line

It is also possible to launch directly the Bart GUI from a command line. It gives the user more possibilities, as launching the GUI without parameters and setting them graphically, providing more rule files, customizing the way to look for seen machines …

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

This section describes how the Bart GUI can be launched from a command line.

II.2.1 Command usage

Bart GUI command line must be launched as follow:

```
bartgui [options] [component]
```

"component" is the path to the component to refine.

Here is the list of options that can be passed to the GUI from the command line:

Parameter	Comment
-r rmf_file	Adds "rmf_file" as a rule file
-I directory	Adds the given directory to the list of directories searched for machine files
-p	Name of the project that should be loaded (requires -b)
-b	Path to the bdp of the project
-a	Visibility file
-P	Path of bart external command
-h	Displays this help message

Figure 1 : Bart GUI command line parameters

II.2.2 Input files

As an input, Bart GUI must be given on the command line at most one component to refine.
It also can be given zero, one or several Bart rule files.

All these inputs are not mandatory, as they can be selected or modified directly from the GUI.

II.2.3 Visibility for loaded component

When Bart GUI must load seen machines, given component abstraction or definition files, it must be able to find their associated files on the file system. So at the command launching user may provide necessary information. There are three ways to do this:

- *-I dir:* This option allows the user to directly specify directories components to load must be searched in. So there can be several –I parameters on command line.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

- *-a file_name:* This is used to give Bart a visibility file. Each line of this file is a research directory. This option could be used together with –I option, in this case file directories and command line directories are added

- *-b path and –p project:* With these options, information about an AtelierB project is provided for searching components. –p option indicates the project name, and –b is the project bdp path. –b and –p must be present together.

All these options can not be used at the same time. Only AtelierB project resolving is used if all these parameters are given on command line.

Values for these parameters can also be set directly in the GUI once it has been launched.

II.3 Advanced users – Selecting a new component to refine directly in the Bart GUI

As said in previous sections, the Bart GUI can be launched with a given component to refine. But it is also to possible, once the GUI has been started, to select a new component to refine by choosing "Open" entry in "File" menu.

All settings that were already present in the GUI, such as rule files of the list or directories for seen machines, are not erased when a new component to refine is selected from the GUI. There may be non necessary rule files or other settings that could impact the new refinement.

So when using this method, user should be careful, at the new refinement beginning, that all settings are as expected.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

III GUI OVERVIEW

This section is an overview and short description for each graphical part of the GUI. Other information will be given further on specific parts for their usages.

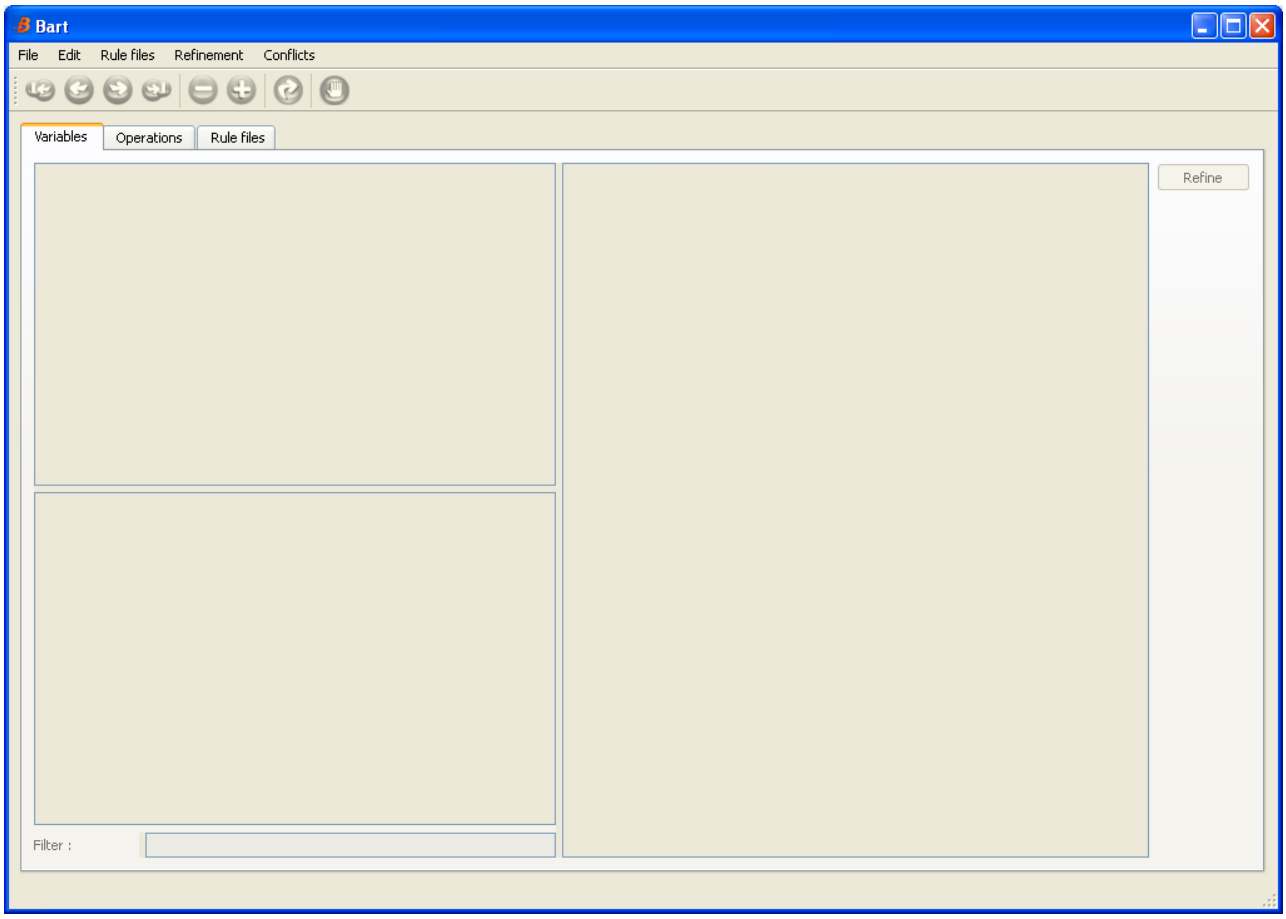


Figure 2 : Bart GUI

Figure 2 shows the Bart GUI when it is launched with no component to refine. In this case most of its buttons are inactivated.

III.1 Menu bar

In this section we'll give a short description of Bart GUI menu entries

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

III.1.1 File menu

Entry	Description
Open	Allows selecting a .ref or .mch file that contains the new component to refine. Same as dropping a component file anywhere (except in specific dropping areas) from outside the GUI. Interactive refinement currently processed, if any, is lost. Variables refinement is automatically processed for the new selected component
Quit	Close the Bart GUI

Action 1 : File menu

III.1.2 Edit menu

Entry	Description
Sees directories	Open a frame managing information about directories seen components must be searched in (only for current GUI session)
Preferences	Allows to manage GUI preferences

Action 2 : Edit menu

III.1.3 Rule files menu

Entry	Description
Add rule file	Add of a rule file for current interactive refinement

Action 3 : Rule files menu

III.1.4 Refinement menu

Entry	Description
Refine variables	Restart interactive refinement from variables refinement, without reloading the component to refine

Action 4 : Refinement menu

III.1.5 Conflicts menu

Entry	Description
Show conflicts	Let the user choose a .xml file, generated by Bart automatic refiner in case of deadlock, that must be graphically displayed by the GUI.

Action 5 : Conflicts menu

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

III.2 Action bars

This section provides a short description for each button of the Bart GUI action bar.

Figure 3 : Bart GUI action bar

Some of the buttons are relevant and activated only in certain parts of refinement process.

This action bar is divided in four sections that are described here.

III.2.1 Refinement control buttons

These buttons can only be activated when operations refinement tab is selected. They are then activated according to the current step of refinement (cf. VII for more details)

Button	Name	Description
	Reset	Resets current operation refinement. Activated only if one step at least has been processed
	Backward	Unrefines deepest refined subnode of currently selected one. This action may be useful to quickly get back of several steps in a branch refinement, but, for a better control of unrefinment, user should use the "double click on refined node" action (cf. VII.3.2.3) Activated only if one step at least has been processed
	Forward	Processes currently selected node, or its first unrefined subnode (cf. VII) Activated only if such a node is present
	Complete refinement	Processes completely currently selected operation refinement Activated if the tree still contains at least one unrefined node

Action 6 : Action Bar – Operation refinement control

III.2.2 Zoom buttons

These buttons can only be activated when operations refinement tab is selected. They are then activated according to previous uses of zoom functionality.

Button	Name	Description

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

	Zoom	Zooms on a node of refinement tree
	Unzoom	Gets back of one step in the zoom stack. Activated only if a zoom has been performed before.

Action 7 : Action bar - Zoom

III.2.3 Refresh button

Button	Name	Description
	Refresh	Restarts refinement from the beginning. Like menu entry *"Refinement > Refine variables",* this button restarts refinement from variables refinement. But unlike this action, it also reloads the component to refine, so it is useful when modifications in the component file have been done outside the GUI

Action 8 : Action bar - Refreshing

III.2.4 Stop button

Button	Name	Description
	Stop	Stops current background process, if any Some operations of the GUI are realized in background: complete refinement of an operation, or initialization of operations refinement status. These processes may sometimes be long or turn into infinite loops, if bad rules were written. This button, activated only if such a process is currently running, allows the user to interrupt it. The stopping state depends on which process was stopped, and will be described in further sections.

Action 9 : Action bar - Interrupting

III.3 Variables tab

This section of the GUI concerns variables refinement. It is activated only if a component to refine was selected.

It shows which rule was selected for each variable, or which variables could not have been refined.

As variable refinement is a simple process (here it means non recursive as substitution refinement), this tab shows the result of variable rule research, but doesn't permit to control it directly, as tactics and user passes are already present in this purpose.

If a variable refinement error occurs, user should modify his rules and reset variable refinement from this tab.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

III.4 Operations tab

This section of the GUI is about the refinement of operations. Refinement of operations also includes refinement of INITIALISATION clause, which is simply present as "INITIALISATION" in the list of operations to refine.

The tab presents a refinement tree for each operation, and permits to navigate in it. The user can modify substitution rules during the refinement in order to correct refinement errors that may have occurred.

III.5 Rule files tab

This part is a rule file selector and editor for the GUI. It contains the list of the rule files currently used for interactive refinement. At start, this list contains rule files provided by AtelierB or on the command line, and preloaded rule files set in preferences (cf. IV.2.2).

Rule files can be added to the current list from this tab. Furthermore, rule files can be edited from here.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

IV GUI SETTINGS

This section presents which settings can be modified by the user in the Bart GUI. All these settings are accessible by the menu "Edit".

IV.1 Environment directories

To access this panel, the user must choose "Edit > Sees directories".

It presents and permits to modify the current mode for searching files of component to refine environment (seen machines, abstraction).

The user can switch between research modes by checking or unchecking suitable box:

- Usage of an AtelierB project information
- Searching in a list of directories

User can also add searching directories by using suitable button or dropping them into the frame, and remove some directories. Order of research directories can be modified by moving a line in the list.

These settings describe currently used mode and parameters, and are only relevant for current session of Bart GUI, so they will not be kept unchanged after a restart. Their value at the GUI start depends on which parameters were used to launch the GUI in AtelierB or from command line.

As environment information is only used at component loading, these parameters modification are only considered when a new component to refine is selected directly in the GUI (cf. II.3).

Note: If option –a (visibility file) was given as a research mode on command line, list of directories will contain all the ones listed in the file

IV.2 Preferences

This section describes preferences in Bart GUI. Each subsection corresponds to a tab of preference frame accessible by "Edit > Preferences".

IV.2.1 Commands

This panel permits to modify path for accessing Bart and DOT command which are used by the GUI.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

This should be used only by advanced users, in case of problem, as these values are normally preset.

IV.2.2 Rule files

This panel allows setting a list of rule files that will be loaded at each GUI start. They will be added to the ones provided at launch by AtelierB or on the command line.

Files can be added to the list with the suitable button or by drag and drop from outside the GUI, or be removed from the list.

Order of the files can be modified by moving a line in the list.

Furthermore, the user can choose to load rule file associated to given component each time that a new refinement begins (and not only at launch by AtelierB or on command line).

IV.2.3 Directories

This panel allows setting a list of directories that will be automatically added, at each start, to the list of searching directories (this list can be then accessed in "Sees directories" frame, cf. IV.1).

Directories can be added to the list with the suitable button or by drag and drop from outside the GUI, or be removed from the list.

Order of directories can be modified by moving a line in the list.

This preference is only relevant when the GUI uses directories list as a searching mode.

IV.2.4 Display

This panel provides functionalities to modify display, as user can modify some of the colors used in some parts of the GUI.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

V METHOD OF INTERACTIVE REFINEMENT

This section will present how and when interactive refinement should be used.

V.1.1 When is interactive refinement necessary?

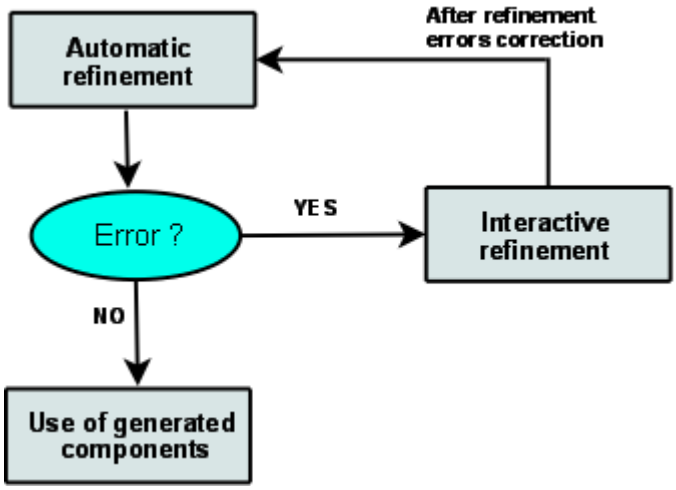


Figure 4 : Usage of interactive refinement

As said before, Bart GUI doesn't produce output components of the refinement. It can only be used to see which rules were used to refine each element, and to show refinement errors.

So interactive refinement is only used if an error occurred in automatic refinement, or automatic refinement was successful but its result was not what the user expected.

After correcting rules and / or model in the GUI in order to correct refinement errors and obtain a suitable result, automatic refinement must be relaunched to finally generate output refined components.

V.1.2 Interactive variable refinement

When a component to refine is given, Bart GUI automatically processes variables refinement. The GUI must be able to refine successfully variables for operation refinement tab to be activated.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Result of variable refinement is accessible in "Variables" tab. If an error occurred, hypothesis stack is available to check if expected rules were not applied because of a lack of hypotheses.

According to information provided by variables refinement tab, user should modify the model, in order to add hypotheses, or modify or add more specific refinement rules for problematic variables.

If model was modified, user can relaunch the variable refinement with the "Reload" button (cf. III.2.3), as the component to refine and its environment must be reloaded. But in this case, it should be better to close the GUI, type-check again the modified components in AtelierB and relaunch the GUI.

If only rules were modified, there is no need to reload the component. So user can simply relaunch variables refinement.

This process must go on as long as there remain some variables that could not be refined.

V.1.3 Interactive operation refinement

Once variables refinement is successful, the GUI can be used to handle errors in operations refinement, if any.

The user must then select one of the operations that could not be refined in the operation tab. The tab provides functionalities to process and display the operation refinement tree.

So user can reach nodes of the tree in which an error occurred. For each one of these nodes, hypothesis stack, syntactically matching rules and more information can be displayed. Using this information, rules can be written or modified in order to make this node successfully refined. Then the refinement for this operation can be processed further.

For each operation that could not be refined, the user should find every node where an error occurs, and add or modify rules to make them successfully refine.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VI INTERACTIVE REFINEMENT OF VARIABLES

This section will describe interactive refinement of variables through presenting parts of the "Variables" tab of the GUI.

VI.1 Consulting found variable rules - Variables rule tree

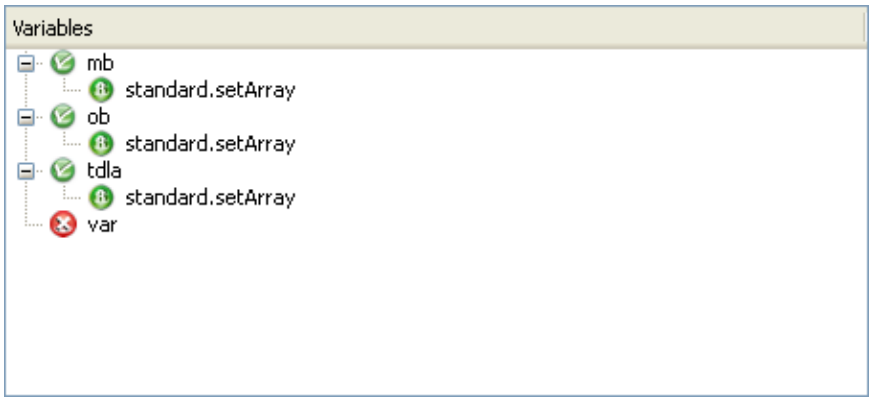


Figure 5 : Variables rule tree

This view presents, once a component has been selected and variables refinement processed, which rule has been selected for each variable.

The result is tree-shaped. Variables that could successfully be refined are marked with the success icon (, as mb, ob and tdla in Figure 5). Each node for successfully refined variable has its selected rule as a subnode (standard.setArray in Figure 5).

Variables that could not be refined are marked with the red error icon (, as var in Figure 5).

Action	Effect
Right click on the tree > Expand all	Expands completely the variable rule tree
Right click on the tree > Collapse all	Collapses completely the variable rule tree
Double click on a rule name	Displays the selected rule in the rule display panel

Action 10 : Variable rules tab

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VI.2 Displaying a selected rule – Variable rule display panel

```
RULE setArray
VARIABLE @a
TYPE
  SETARRAY(@a , @b , @c)
WHEN
  @a : POW(@b) &
  (ENUM(@b) &
  match( @b , @c) or (ABCON(@b) or COCON(@b) &
  @b : POW(@c) &
  SET(@c)))
IMPORT_TYPE
  @a <: @c
CONCRETE_VARIABLES
  @a_i
INVARIANT
  @a_i : @c --> BOOL &
  @a = @b /\ (@a_i~)[ {TRUE}]
END
```

@c = t_block_i

Figure 6 : Example of variable rule display panel content

This panel shows the rule that has been selected in the variable rules tab. For example, Figure 6 is the content of the panel if the user has clicked on node "standard.setArray" in Figure 5.

If no rule has been selected, the panel displays nothing.

It also permits to see values of jokers that made the rule selected.

Action	Effect
Pointing on a joker in the panel	Displays the value of the pointed joker

Action 11 : Variable rule display panel – Joker value displaying

Figure 5 shows an example of joker value displaying. In this case user pointed on @c.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VI.3 Consulting hypothesis for refinement - Hypothesis panel

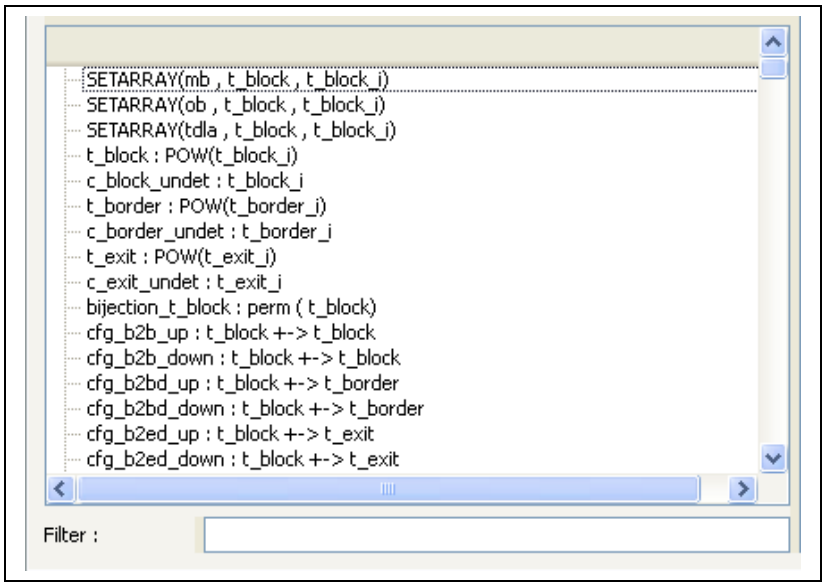

SETARRAY(mb , t_block , t_block_i)
SETARRAY(ob , t_block , t_block_i)
SETARRAY(tdla , t_block , t_block_i)
t_block : POW(t_block_i)
c_block_undet : t_block_i
t_border : POW(t_border_i)
c_border_undet : t_border_i
t_exit : POW(t_exit_i)
c_exit_undet : t_exit_i
bijection_t_block : perm (t_block)
cfg_b2b_up : t_block +-> t_block
cfg_b2b_down : t_block +-> t_block
cfg_b2bd_up : t_block +-> t_border
cfg_b2bd_down : t_block +-> t_border
cfg_b2ed_up : t_block +-> t_exit
cfg_b2ed_down : t_block +-> t_exit
Filter :

Figure 7 : Hypothesis panel

This panel shows hypotheses present in Bart stack when the variables refinement is done.

Displayed hypothesis can be filtered by selecting hypothesis types to display, or by specifying a string that must appear in displayed hypothesis.

VI.3.1 Filtering hypothesis by type – Filters menu

A menu permits, from hypothesis list, to filter hypothesis by type.

Action	Effect
Right-click in the list > Filters	Pops-up a window proposing to filter hypothesis by type
Right click in the list > Reset filters	Disables all previously selected filters Activated only if some filters were previously selected

Action 12 : Hypothesis panel – Filter menu

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

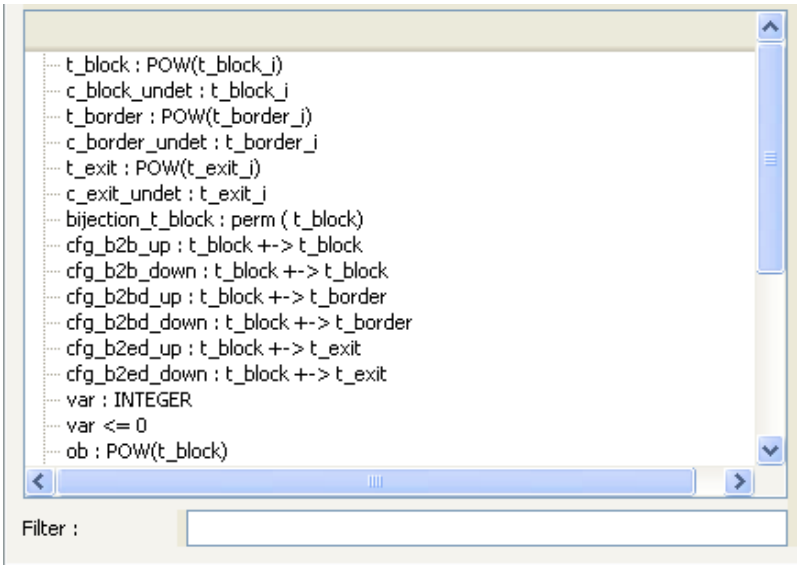


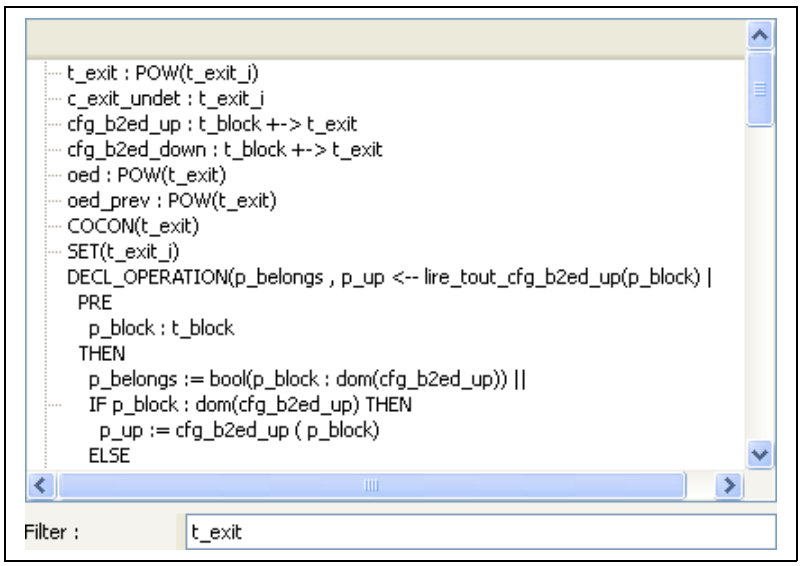
Figure 8 : Hypothesis stack from figure 7, without guards and type predicates

VI.3.2 Filtering hypothesis by content – Filter text bar

The text bar in the hypothesis panel lets the user displays only hypothesis containing given string. The filtering doesn't require any validation to be done, as content of the panel is updated each time the text bar content is updated

Action	Effect
Modify content of text bar	Updates list of hypothesis with only those which contains the new value of the text bar

Action 13 : Variable hypothesis filter text bar



Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Figure 9 : Hypothesis stack from figure 7, filtered with t_exit identifier

Ce document est la propriété de ClearSy - Toute reproduction ou utilisation partielle ou totale de celui-ci est interdite sans son accord préalable.

VII INTERACTIVE REFINEMENT OF OPERATIONS – OPERATIONS TAB

This section will show how operations can be refined using the GUI, through presenting different parts of the operation refinement tab.

The Bart GUI operation refinement tab is activated only if all following conditions are true:

- A component to refine has been selected
- Selected component had no variable to refine OR all variables could be successfully refined
- There is at least one operation to refine OR selected component has an initialisation clause

The underlying structure of operation refinement is, as said in [R1], a tree. The aim of the GUI operations tab is to show this tree for operations to refine, in order to see which rules have been selected, or, if so, which nodes of the tree could not be refined.

Section VII.3 will more precisely describe how Bart GUI presents the refinement tree, and the different ways provided to the user to navigate in it.

VII.1 Selecting an operation - Operation list view

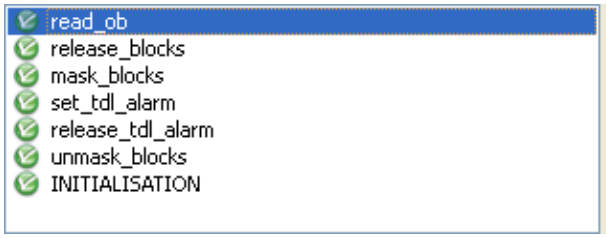


Figure 10 : Operations list view

This panel presents all operations to refine for the selected component, including the content of INITIALISATION clause, if any.

At the selection of the component to refine, or after a successful variable refinement, if operations tab must be activated (see conditions before), Bart GUI launches an automatic refinement on each operation in order to determine its status.

Refinement status can be success (, operation could be completely automatically refined by the GUI), error (, an error occurred during GUI

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

automatic refinement for this operation), or undefined (⚠, automatic refinement in GUI was interrupted before status could be computed for this operation).

Initial automatic refinement in GUI is launched in background. If the process is too long or becomes an infinite loop due to bad written rules, the user can interrupt it using "Stop" button (cf. III.2.4). In this case, all operations that could not be processed at the initial automatic refinement will have the "undefined" status.

After the initialization step, operations status may be modified by user processing of operation rule tree, if refinement can be completed of if new errors occur, for example.

This view is used to switch between operations to refine.

Action	Effect
Left-click on an operation name	Selects clicked operation as the new operation to refine. All others parts of operation refinement tab are updated according to this modification

Action 14 : Operations list

When a new operation to refine is selected, current refinement information, if any, is lost. Going back on the previously selected operation will restart its refinement from the beginning.

VII.2 Current refinement status – Refinement status bar

This bar shows information on the state of current operation refinement.

Figure 11 : Refinement status bar

VII.2.1 Operation name

This is the text field at the bottom-left of the bar. It shows which operation is currently refined.

If the user clicks in current operation tree on a node that is in fact a part of a new imported operation refinement (cf. VII.3.1), the name of the sub operation will be displayed in this field.

VII.2.2 Refinement type

This is the text field at the top-left of the bar. It shows which type of refinement is processed for current operation. The two possible values for this filed are (cf. [R1] for differences between structural and regular refinement):

- Structural refinement
- Regular refinement

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Refinement mode is automatically selected when the user selects a new operation to refine in operations list view.

Operations that don't contain structural substitution (cf. [R1]) are refined in regular mode.

Structural refinement is automatically started for operations that contain structural substitutions. At the first refinement error that occurs in structural refinement mode, the GUI automatically switches to regular one. In this case, a refinement reset with "Reset" button (cf. III.2.1) only restarts the regular refinement from the beginning.

In this particular case, if the user wants to restart in order to correct and complete structural refinement, the operation should be selected again in operations list view, or the machine to refine should be reloaded.

VII.2.3 Refinement status

This is text field at the top right of the bar. It shows the refinement status for currently refined operation.

The possible values for this field are:

- Processing refinement: There still are unrefined nodes in the tree, and no error has occurred yet.
- Refinement successful : The operation could be completely refined without errors
- Error occurred in refinement : At least one error occurred during this operation refinement, but there are still unrefined nodes in the tree
- Errors, no more branches: At least one error occurred during this operation refinement, and there are no more refinement nodes to proceed.

VII.3 Processing and navigating in an operation refinement

VII.3.1 Bart GUI refinement tree

VII.3.1.1 Presentation

As described in [R1], refinement of operations can be represented by a tree, as the process is recursive.

Consequently, one of the main functionality of the Bart GUI is to present this refinement tree for each operation.

Each node of this tree is associated to a substitution to refine, and after its treatment, if no error occurs, to a refinement method (rule or predefined behavior, cf. [R1]) and a refinement result. So each node represents the refinement process of a certain substitution.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VII.3.1.2 Content

Each node of the tree can have as subnodes:

- Refinement result of current rule, if it had a REFINEMENT clause.
- Substitutions coming from application of a predefined refinement behavior (as inside substitution of a block or guarded substitution, or branches of parallel or semicolon)
- Content of SUB_REFINEMENT clauses
- Top level substitutions of imported operations defined by this node's refinement. This is specific to the GUI, in order to show the whole refinement of an operation (including its newly defined imported operations) in a unique tree.

Rules with IMPLEMENTATION clauses are terminal (as said in [R1] their result is not refined again), but this kind of node in the GUI can still have sub nodes, as the rule could have introduced imported operations or contain SUB_REFINEMENT clauses.

As said in [R1], rules with REFINEMENT clause can also have SUB_REFINEMENT clause. In this case, the result of REFINEMENT can not be instantiated as long as SUB_REFINEMENT was not treated. In the GUI, this kind of node will have as subnodes both content of SUB_REFINEMENT and REFINEMENT (cf. VII.3.2).

As all imported operations defined by the one currently refined are presented in a unique tree, it might get quite voluminous. In order to simplify treatments, the user may use the zoom functionality.

VII.3.2 Consulting the refinement tree - Refinement tree view

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

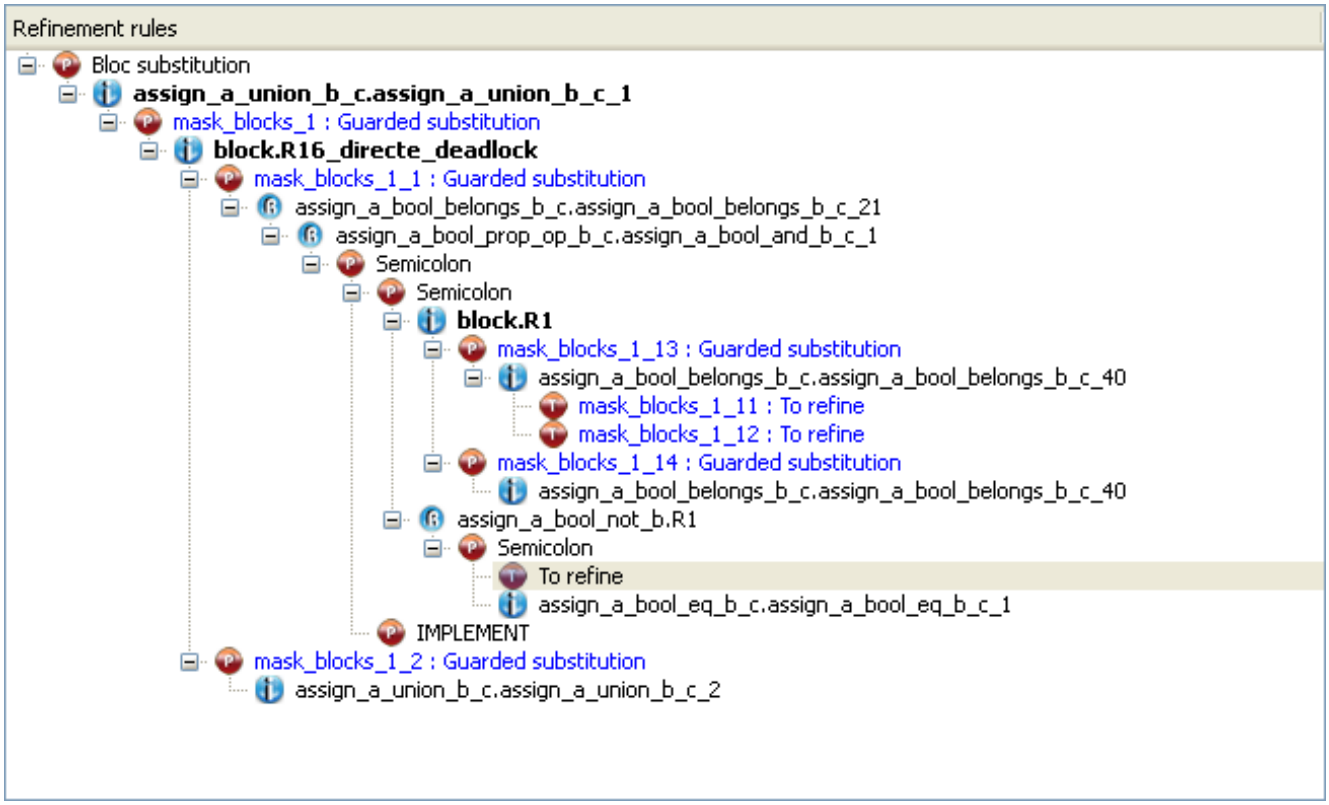


Figure 12 : Example of an operation refinement rule tree

VII.3.2.1 Icons

Icon	Meaning
	Still unrefined node
	Node refined using a Bart predefined refinement behavior
	Node refined by a substitution rule with an IMPLEMENTATION clause
	Node refined by a substitution rule with a REFINEMENT clause
	Node with a substitution that could not be refined

Figure 13 : Meaning of icons used in operation rule tree

VII.3.2.2 Formatting information

Some refinement information is provided to the user through the presentation of text associated to refinement tree nodes.

Formatting	Meaning
Node text	The rule found for this node has introduced new imported operations
Node text	Substitution of current node is the top-level substitution of a new imported operation
Node text	Substitution of current node is the content of a SUB_REFINEMENT clause
Node text	Inactivated node

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

	This kind of node is a REFINEMENT clause content of a rule that also contains SUB_REFINEMENT clause, which has not been fully refined in the GUI for now. So substitution to refine for this node can not be computed for now.

Figure 14 : Meaning of formatting used in operation rule tree

VII.3.2.3 Processing refinement – Operation refinement tree nodes

The user has the possibility to process refinement by directly manipulating nodes of the operation refinement tree.

Action	Effect
Click on a node	Modify currently selected node. Elements described in VII.3.4 are updated
Double-click on an unrefined node (with icon)	Processes refinement of selected node, i.e. searches a rule or a predefined refinement behavior to refine its associated substitution. Refinement status bar may be updated after this action according to the result (error, end of refinement…). Associated node icon is also updated according to the result. Suitable subnodes are added to the node, if necessary (SUB_REFINEMENT clauses, REFINEMENT clause, new imported operations, predefined behavior), and the refinement result panel is updated consequently.
Double-click on a refined node (with icon different of)	Cancels the refinement of this node and its subnodes. The node comes back to unrefined status, and all its subnodes are deleted. Refinement status bar may be updated after this action (errors can disappear, refinement may be uncompleted). The refinement result panel is updated after this action

Action 15 : Operation rule tree nodes

In the Bart GUI, refinement of nodes can be processed in any order. Current branch doesn't need to be fully refined for the user to try another one.

As said in VII.3.1 and VII.3.2.2, some rules can contain both REFINEMENT and SUB_REFINEMENT clauses. In this case, node representing REFINEMENT clause content is inactivated as long as SUB_REFINEMENT clause has not been completely successfully processed in the GUI.

For this kind of node, if an unrefinement action is processed in the fully refined SUB_REFINEMENT clause content, node of REFINEMENT clause will become back inactivated.

VII.3.2.4 Using the tree – Operation refinement rule tree context menu

This section presents actions that can be processed by using the context menu of operation refinement rule tree, which can be accessed by a right-click on a tree node.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Action	Effect
Right click on tree > Expand all	Expands all the nodes of operation rule tree
Right click on tree > Collapse all	Collapses all the nodes of operation rule tree
Right click on tree > Zoom	Zooms on currently selected node. Same as zoom button in action bar (cf. III.2.2)
Right click on tree > Unrefine	Unrefines currently selected node (node must have been refined before). Same as a double-click on the node
Right click on tree > Show result	Allows to isolate current refinement result of this node and its subnodes in the refinement result panel by highlighting it
Right click on tree > Next	Refines currently selected node, or its first found unrefined node, if any. Same as using "Next button" (cf. III.2.1) on current node, or as double click on the node, or its first unrefined subnode
Right click on tree > Previous	Unrefines deepest refined subnode of currently selected one. Same as using "Backward" button (cf. III.2.1)
Right click on tree > Go to	If current node in a node successfully refined with a rule, displays this rule in the "Rule files" tab

Action 16 : Operation refinement tree context menu

VII.3.3 Consulting the refinement result – Refinement result panel

```
/* ?1_1 := bool(cc_tdl_ack = TRUE)? */
[par_out_0_1:=1_1] <-- release_tdl_alarm_1=
    [
      PRE
        0 = 0
      THEN
        par_out_0_1 := bool(cc_tdl_ack = TRUE)
      END
    ] ;
IF 1_1 = TRUE THEN
  tdla := {}
END
```

Figure 15 : Example of refinement result panel

VII.3.3.1 Content

The refinement result panel shows the current result produced by the operation refinement.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

It depends on the state of the rule tree. Each node refinement or unrefinement modifies the B code displayed in this panel .

At the beginning it contains the top level substitution of the operation to refine. Then, as user processes nodes refinement, the GUI calculates recursively current result according to found rules and refinement behaviors at this step. At the end of the operation refinement, if it was successful, this panel displays the operation that would be implemented in output components by automatic refinement.

VII.3.3.2 Imported operation calls presentation

As shown on Figure 15, the display panel mainly contains classical B code. But, as for operation refinement tree, if refinement of currently treated operation introduces new operations, their refinement result is shown is the same panel.

This is done using a special presentation for new imported operation calls.

```
[p_out_1 := val_out_1,…] <-- operation_name([p_in_1 := val_in_1, …]) =
[
      Imported operation “operation_name” current refinement result
]
```

Figure 16 : Example of presentation of an imported operation call

The first line of the example presents the imported operation prototype, and values for its parameters in the operation call. This operation call is a part of a result clause of a refinement rule, which contained an IMPORTED_OPERATION substitution. The real generated substitution, in a result component, would be:

```
val_out_1,…  <-- operation_name(val_in_1, …)
```

Code section between [] in Figure 15 is the current refinement result for imported operation “operation_name”.

VII.3.3.3 Controlling refinement with display panel – Highlighted unrefined substitutions

Figure 15 shows some highlighted substitutions. These are parts of current result that have not been refined yet. So each highlighted part is the substitution to refine associated to a still unrefined node of the tree.

Action	Effect
Put mouse pointer on an highlighted part of code	A part of code text changes of color. Sometimes, consecutives lines of code are all highlighted, but in fact correspond to different unrefined nodes. This functionality permits to see clearly correspondence between code and rule tree
Put mouse pointer on a new imported operation	When user points to a parameter name in a new imported operation refinement result, the GUI displays a box that shows effective call value for the parameter (shortcut for notation described in VII.3.3.2)

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

parameter	If there are several levels of nested operations, and if necessary, this box may contain several lines to symbolize all operation calls.
Click on highlighted part of code	Selects the node in rule tree that corresponds to clicked part of code. Same as directly clicking on the node rule tree (cf. VII.3.2.3)
Double-click on highlighted part of code	Refines the node that corresponds to clicked part of code. Same as double-click on the associated node in the tree (cf. VII.3.2.3)

Action 17 : Refinement result display panel

VII.3.4 Node depending tabs

The Bart GUI contains, in the operation tab, several panels with a content that depends on which node is currently selected in the operation refinement tree view.

VII.3.4.1 Hypothesis stack

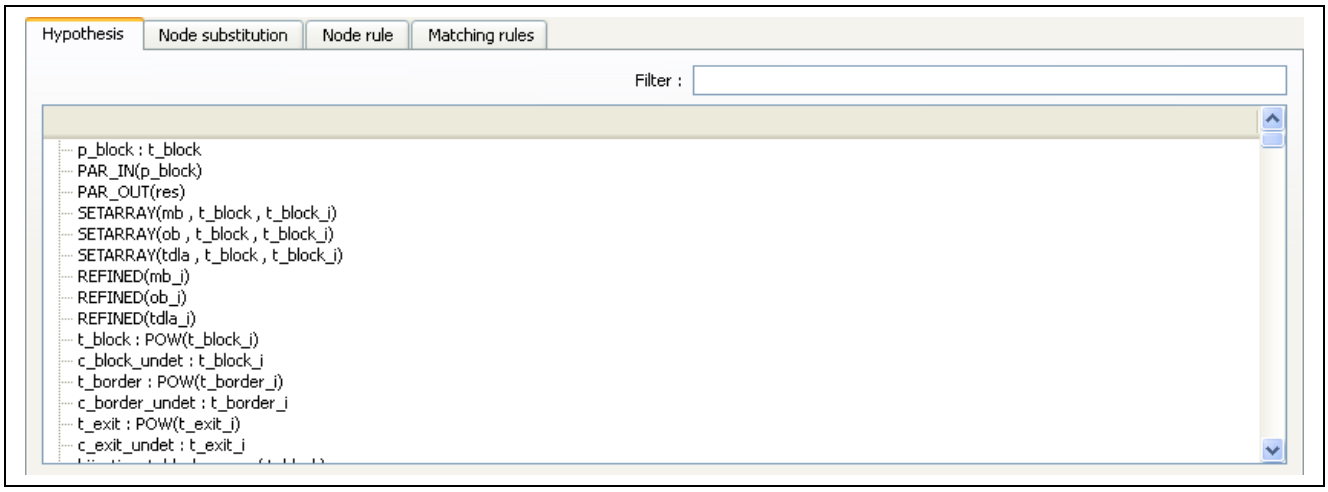


Figure 17 : Hypothesis tab for operation refinement

First node depending panel is the hypothesis stack for operation refinement. As described in [R1], the stack evolves during operation refinement, as hypothesis can be added by refinement of LH substitutions, guarded substitutions, special pragmas…

Hypothesis tab for operation refinement shows the state of the stack for the refinement of the node that is currently selected in the tree.

Action	Effect
-	Actions that can be performed in this panel are the same as for variable refinement hypothesis panel (cf. VI.3)

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Action 18 : Hypothesis tab for operation refinement

VII.3.4.2 Node substitution

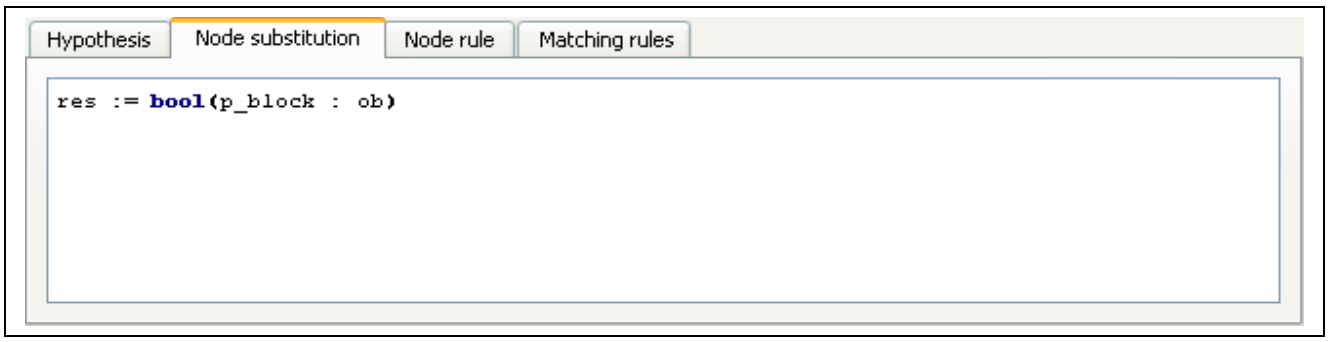


Figure 18 : Node substitution tab

This tab simply displays the substitution to refine associated to the node currently selected in the tree.

VII.3.4.3 Node rule

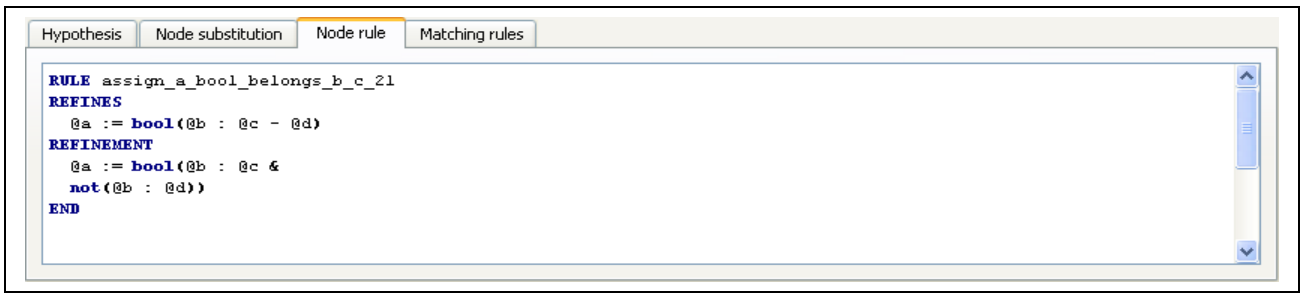


Figure 19 : Node rule tab

This tab displays the substitution rule that was used to refine currently selected node.

If currently selected node is still unrefined, or a predefined behavior was used for its refinement, this tab displays nothing.

Action	Effect
Pointing mouse on a joker	Displays value of the joker, from the instantiation that made the rule be picked

Action 19 : Node rule tab

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VII.3.4.4 Matching rules

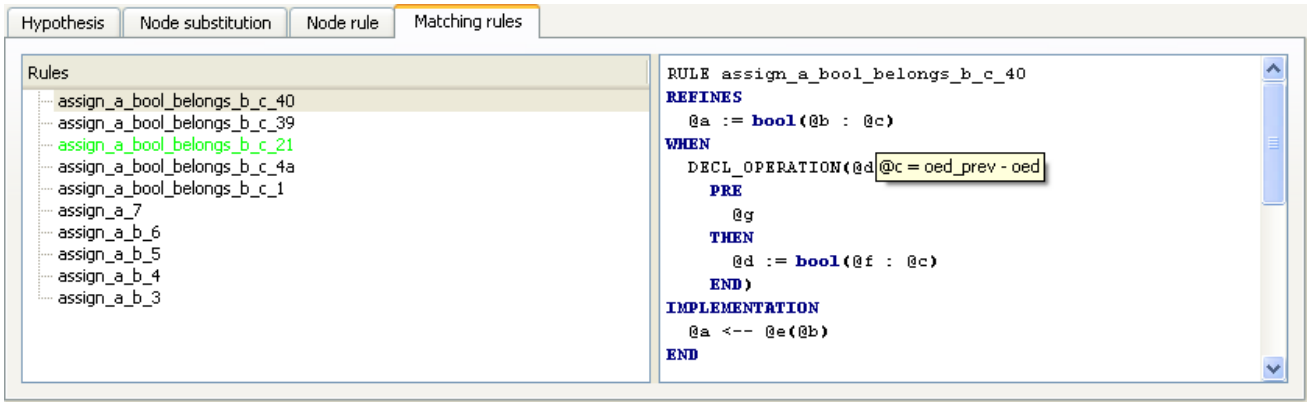


Figure 20 : Matching rules tab

This tab shows all rules with a pattern (REFINES clause) that matches with substitution to refine associated to currently selected node. Only syntactical match of pattern is verified, constraints of listed rules WHEN clauses are not checked.

List of matching rules is built according to the use of user pass or tactic theories, if any, so it is the complete list of rules that can be applied for this node. They are displayed in the same order as they are processed in rule research, i.e. if several rules have a WHEN constraint that can be checked, the lowest in the list will be used.

Matching rules are listed in the left part of the panel, while the right part is used to display them.

If the node refinement has been processed and one of the rules was used, its name is highlighted in green in the list.

Action	Effect
Pointing mouse on a rule name in the tab left part	Displays, in a box, the name of the file that contains pointed rule.
Click on a rule name in the tab left part	Displays the rule in the tab right part
Pointing mouse on a joker in the tab right part	Displays value of the joker, from the instantiation that made the rule match. As only syntactical match of the pattern is checked for building this list, values can be displayed only for jokers present in the rule pattern

Action 20 : Matching rules tab

VII.3.5 Zooming on tree parts

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

As the tree and the result of current operation also contain information about its new introduced imported operations, the display may quickly become very large.

In order to isolate some parts of the tree during refinement, Bart GUI provides a zoom functionality. Zooming means isolating a part of the tree, which temporary root will be currently selected node, and its associated result part.

Zoom and unzoom actions are accessible through the action bar (cf. III.2.2), and operation refinement tree context menu (cf. VII.3.2.4).

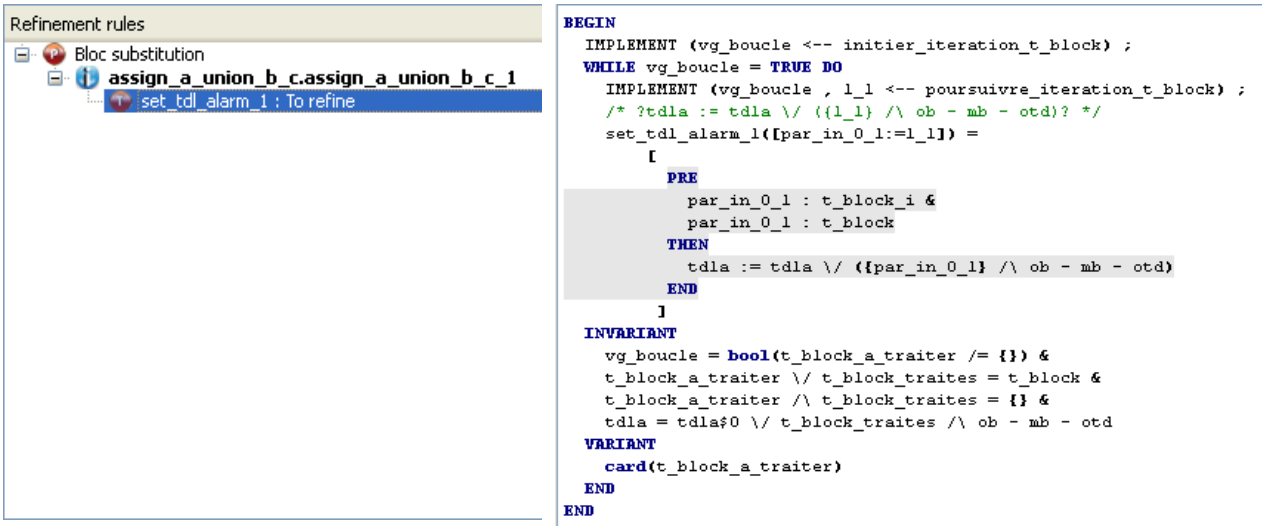
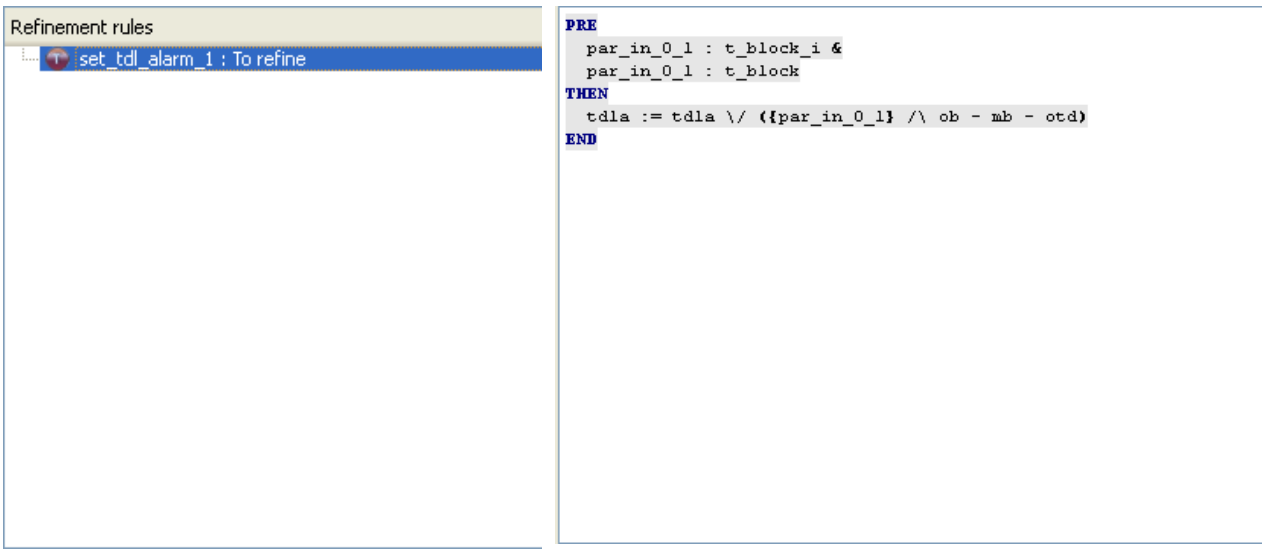


Figure 21 : Refinement tree and result before a zoom action



Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Figure 22 : Refinement tree and result of Figure 21 after zoom on "set_tdl_alarm_1" node

Figure 21 and 22 show effects of a zoom action. Zoomed node becomes root of refinement tree, and only its associated result is displayed in result view. Here in the example, chosen node is a node associated to an imported operation, but zoom actions can be performed on every node of the tree.

Several zoom actions can be performed consecutively. Each one will be memorized, in order for the user to be able to get back to initial state with several unzoom actions.

Unzoom action becomes available after at least one zoom action has been performed. All refinement actions performed in the modified tree or result will be kept when user comes back using unzoom action.

Unzooming means canceling last zoom action, i.e. displaying the more global tree containing current one, and the associated global result.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

VIII ADDING AND EDITING RMF FILES – RULE FILES TAB

In the Bart GUI, user can add and modify Bart rule files in the "Rule files" tab. This section presents its functionalities by describing its sub components.

VIII.1 Managing rule files – Rule files list view

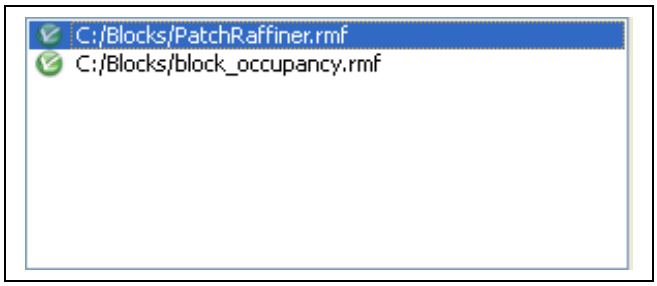


Figure 23 : Example of rule files list view

This list contains rule files actually loaded in the GUI. Each file present in the list has a status, which can be correct file (icon), or file containing errors (icon).

All correct files present in the list are used by GUI during rule research for variable or operation refinement, from bottom to top. Error files are ignored in rule research.

At the beginning, content of the list depends on preferences (preloaded rule files) and files given at launch by AtelierB or on the command line.

After this, user can add other rule files that will be use in following refinements.

When a new component to refine is selected, its eventual rule file is added in last position (most prior) in the list.

Action	Effect
Drag and drop a rmf file from outside the GUI in the list view	Adds a file in the list of rule files used by the GUI. Same as "Add file" button and "Rule Files > Add rule file" menu entry. The file is added in the list at the position it was dropped.
Slide a file from the list to another position	Changes the position of the file in the list.
Right click on the list > "Delete rule file" *OR* *Press "Suppr"*	Removes currently selected rule file from the list. The file is not removed from the file system.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Action 21 : Rule file list

VIII.2 Listing rule file content – Theories tree

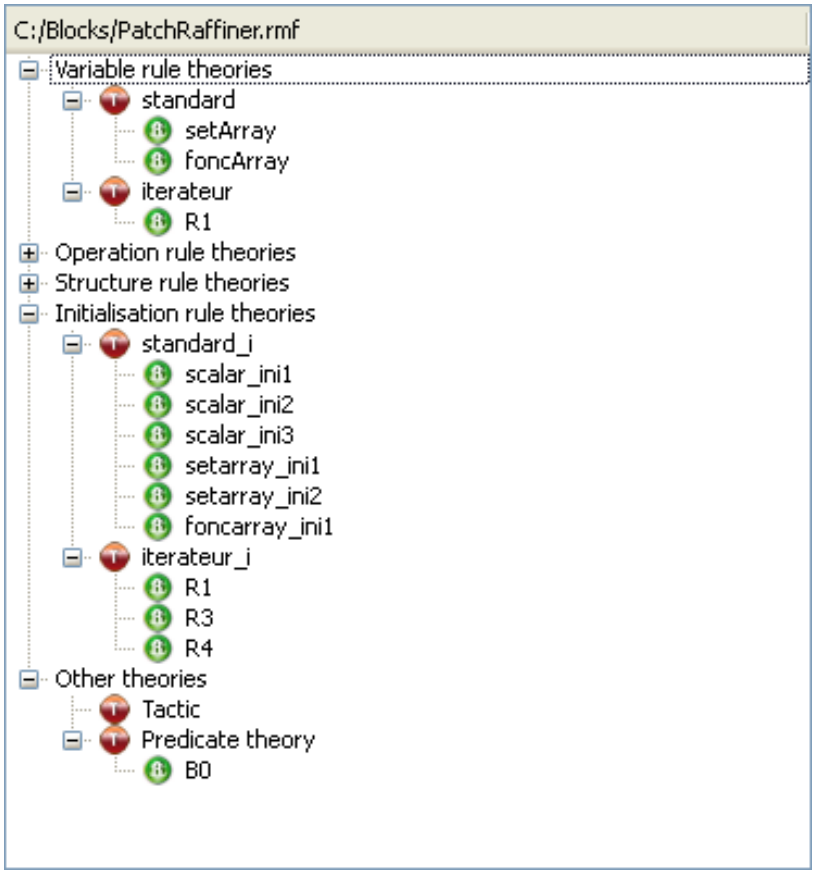


Figure 24 : Example of theories tree for Bart rule base PatchRaffiner.rmf

This view presents a tree-shaped representation of the content of the rule file currently selected in the rule files list.

Theories of represented rule files are gathered by type:

- Variable rule theories
- Operation rule theories
- Structure rule theories
- Initialisation rule theories
- Other theories (Tactic, user pass, predicate theory)

Action	Effect
Right click >	Expands all nodes of the theories tree

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Expand all	
Right click > Collapse all	Collapses all nodes of the theories tree
Double click on a theory or rule name	Reaches the rule or theory in the currently displayed rule file

Action 22 : Theories tree

VIII.3 Consulting rule file content – RMF display tab

This panel is an editor for currently selected rule file. Modifications done in the editor will be considered in next processed refinement steps.

Save of modifications may update the file status in rule files list, as errors may have been introduced or corrected.

Action	Effect
CTRL + S *OR* *"Save" button*	Saves modification. Activated only if previous modifications have been done
CTRL + F	Opens a box which provides some searching functionalities

Action 23 : RMF display tab

VIII.4 Handling rule file errors – Error view

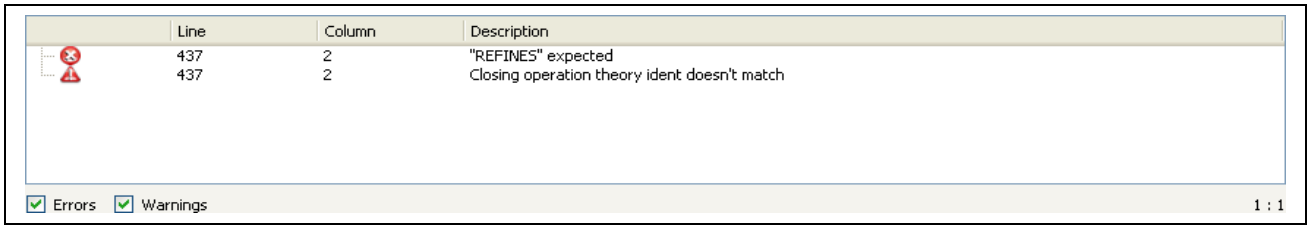


Figure 25 : Rule file editor error view

This frame displays errors (icon) and warnings (icon), if any, from the rule file currently selected in the rule files list.

Content of this frame may be updated after files deleting or modification.

Action	Effect
Click on a line in the error view	Reach the error / warning location in the rule file
Check / Uncheck "Errors" checkbox	Display / Hide errors
Check / Uncheck "Warnings" checkbox	Display / Hide warnings

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

Action 24 : Rule files editor error view

VIII.5 Rule file modifications during interactive refinement

"Rule files" tab is available at any time in the GUI. So user may process actions on rule files when a part of interactive refinement has already been done. Part of the interactive refinement previously processed it still considered correct, even if rules it has used have been modified or deleted, or if new rules that could have applied instead were added.

So generally, if rules had to be modified in GUI in order to correct errors and go further in refinement, the user should check at the end that the whole refinement for this operation is still correct and as expected.

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

IX DEADLOCK RESOLUTION WITH BART GUI

IX.1 Usage with automatic refinement

As said in [R1], Bart automatic refinement may sometimes fail because of a splitting deadlock.

This means that all operations could be successfully refined, but they could not be dispatched in output components. Automatic refinement has reached a step of splitting in which no operation could be implemented. In this case, Bart automatic refinement produces a file named "deadlock.xml" in its output directory.

Interactive refinement through Bart GUI doesn't proceed to splitting, as output components are not generated, but it provides a way to display information contained in the "deadlock.xml" file generated by automatic refinement.

So in case of automatic refinement deadlock on a component, Bart GUI should be only used in this purpose, as no refinement error occurred during rule research. User should apply deadlock information provided in the GUI by modifying its refinement rules, and relaunch an automatic refinement process.

IX.2 Getting deadlock information – Conflicts window

The file "deadlock.xml", generated by automatic refinement can be passed to the GUI in the window accessible through "Conflicts > Show conflicts" menu entry (cf. III.1.5). The GUI then displays a tree structure presenting variables that must be implemented or exported for each operation. From this window, a summarize graph can be generated.

Action	Effect
Click on "Graph" button	Displays a graph summarizing deadlock information. Dot command must be present in the PATH, or its path must have been given in preferences for this functionality to work.

Action 25 : Conflicts information window

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

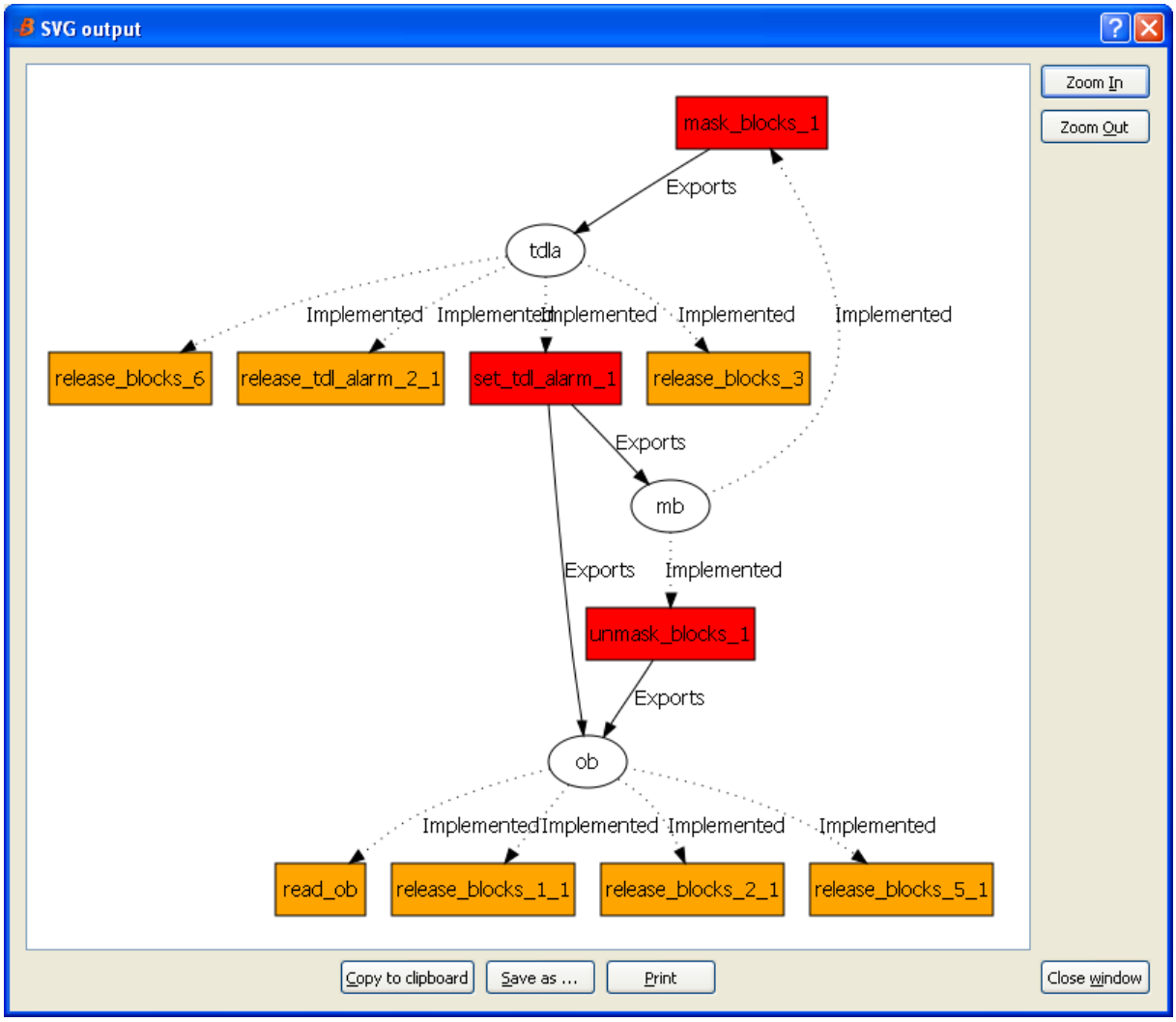


Figure 26 : Example of deadlock information graph

Figure 25 shows an example of deadlock information graph. The symbolism used for these graphs is as follow:

- Ellipses are abstract variables to implement
- Rectangles are operations to implement
  - Orange ones are operations that have been exported from current step component
  - Red ones are operations in conflict, which could not be implemented at this step or exported
- Arrows are links between variables and operations
  - Continuous arrows symbolizes that the operation can be implemented at this step only if the variable is implemented further in the output chain
  - Dotted arrows means that the operation can be implemented at this step only if the variable is implemented too

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

As described in [R1], deadlocks often happen because a cycle is present in operation / variable dependence. These cycles can be graphically seen in deadlock graph between red operations.

Ce document est la propriété de ClearSy - Toute reproduction ou utilisation partielle ou totale de celui-ci est interdite sans son accord préalable.

APPENDIX A: TABLE OF ACTIONS

Action 1 : File menu.....11
Action 2 : Edit menu.....11
Action 3 : Rule files menu.....11
Action 4 : Refinement menu.....11
Action 5 : Conflicts menu.....11
Action 6 : Action Bar – Operation refinement control.....12
Action 7 : Action bar - Zoom.....13
Action 8 : Action bar - Refreshing.....13
Action 9 : Action bar - Interrupting.....13
Action 10 : Variable rules tab.....19
Action 11 : Variable rule display panel – Joker value displaying.....20
Action 12 : Hypothesis panel – Filter menu.....21
Action 13 : Variable hypothesis filter text bar.....22
Action 14 : Operations list.....25
Action 15 : Operation rule tree nodes.....29
Action 16 : Operation refinement tree context menu.....30
Action 17 : Refinement result display panel.....32
Action 18 : Hypothesis tab for operation refinement.....33
Action 19 : Node rule tab.....33
Action 20 : Matching rules tab.....34
Action 21 : Rule file list.....38
Action 22 : Theories tree.....39
Action 23 : RMF display tab.....39
Action 24 : Rule files editor error view.....40
Action 25 : Conflicts information window.....41

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.

APPENDIX B: TABLE OF FIGURES

Figure 1 : Bart GUI command line parameters.....8
Figure 2 : Bart GUI.....10
Figure 3 : Bart GUI action bar.....12
Figure 4 : Usage of interactive refinement.....17
Figure 5 : Variables rule tree.....19
Figure 6 : Example of variable rule display panel content.....20
Figure 7 : Hypothesis panel.....21
Figure 8 : Hypothesis stack from figure 7, without guards and type predicates. 22
Figure 9 : Hypothesis stack from figure 7, filtered with t_exit identifier.....23
Figure 10 : Operations list view.....24
Figure 11 : Refinement status bar.....25
Figure 12 : Example of an operation refinement rule tree.....28
Figure 13 : Meaning of icons used in operation rule tree.....28
Figure 14 : Meaning of formatting used in operation rule tree.....29
Figure 15 : Example of refinement result panel.....30
Figure 16 : Example of presentation of an imported operation call.....31
Figure 17 : Hypothesis tab for operation refinement.....32
Figure 18 : Node substitution tab.....33
Figure 19 : Node rule tab.....33
Figure 20 : Matching rules tab.....34
Figure 21 : Refinement tree and result before a zoom action.....35
Figure 22 : Refinement tree and result of Figure 21 after zoom on "set_tdl_alarm_1" node.....36
Figure 23 : Example of rule files list view.....37
Figure 24 : Example of theories tree for Bart rule base PatchRaffiner.rmf.....38
Figure 25 : Rule file editor error view.....39
Figure 26 : Example of deadlock information graph.....42

Ce document est la propriété de ClearSy - TOUTE REPRODUCTION OU UTILISATION PARTIELLE OU TOTALE DE CELUI-CI EST INTERDITE SANS SON ACCORD PRÉALABLE.